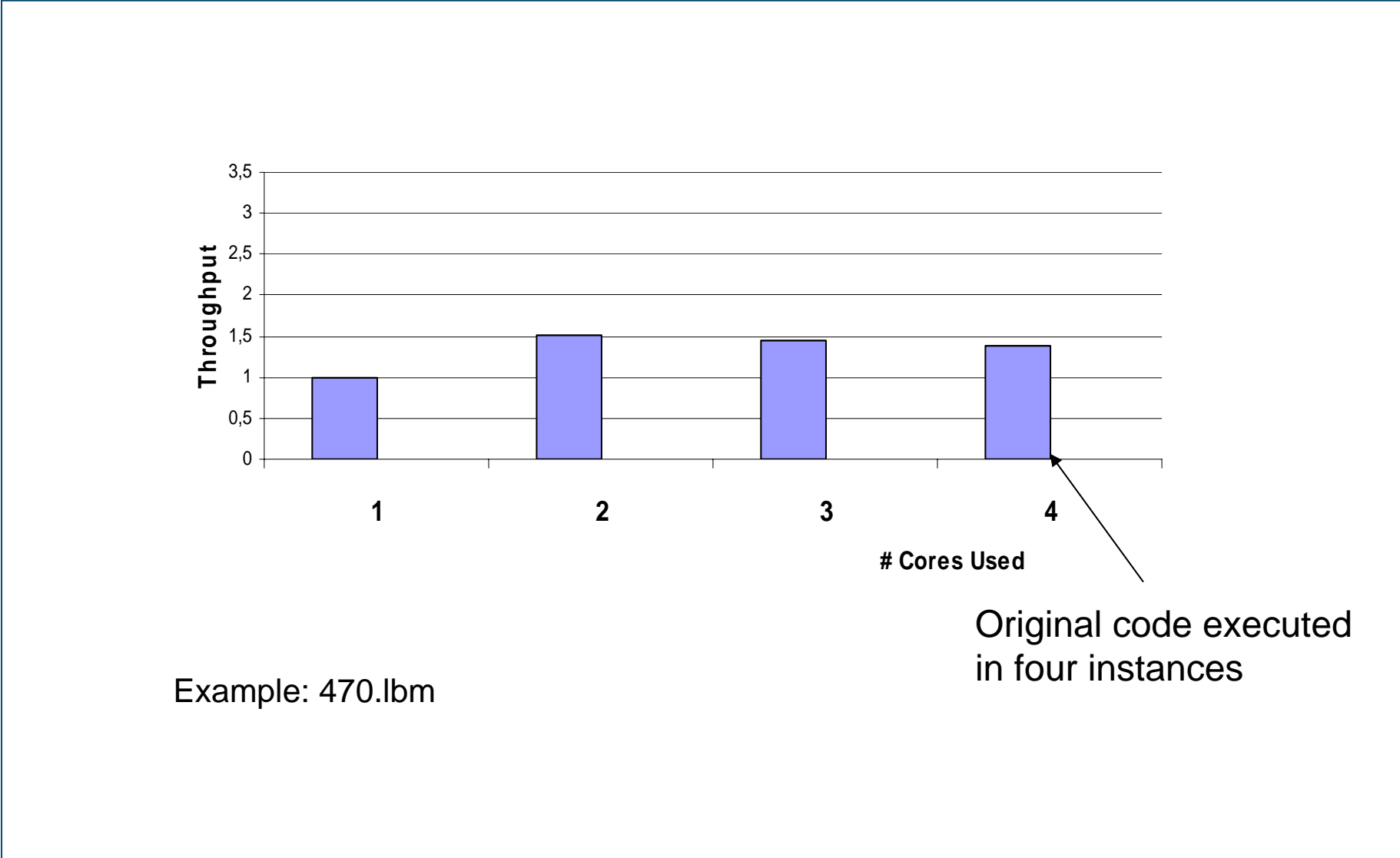




Erik Hagersten, CEO

Also: Uppsala University
SWEDEN

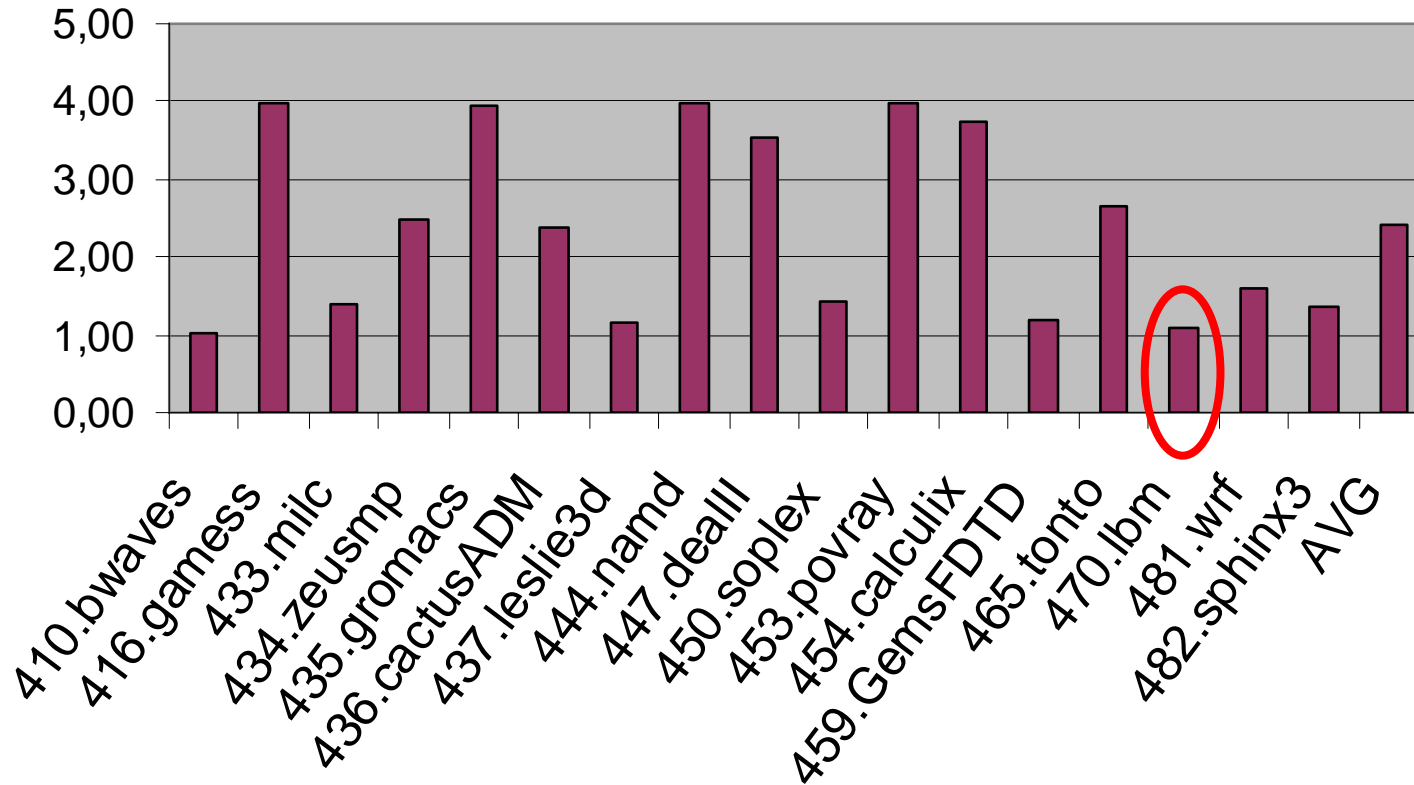
Poor Throughput Scaling?



How Common is Poor Throughput?



Increased throughput on four cores compared with one core, SPECfp



Acumem Technology Tools



Source code

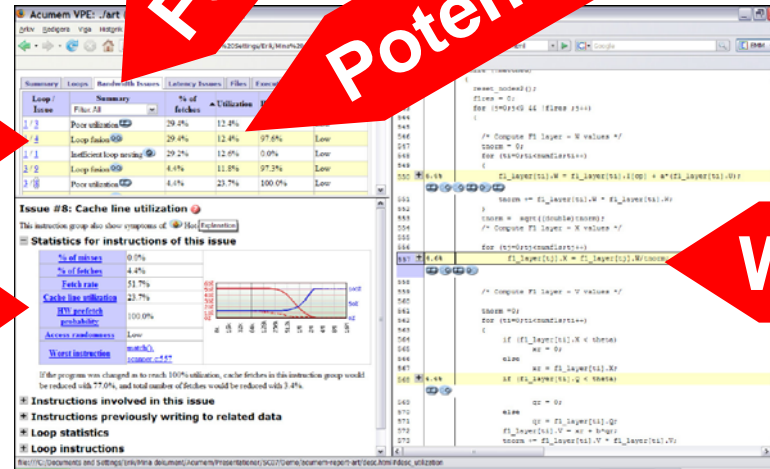
```

/* Unoptimized Array Multiplication: x = y * z  N = 1024 */
for (i = 0; i < N; i = i + 1)
for (j = 0; j < N; j = j + 1)
{
    r = 0;
    for (k = 0; k < N; k = k + 1)
        r = r + y[i][k] * z[k][j];
    x[i][j] = r;
}

/* Unoptimized Array Multiplication: x = y * z  N = 1024 */
for (i = 0; i < N; i = i + 1)
for (j = 0; j < N; j = j + 1)
{
    r = 0;
    for (k = 0; k < N; k = k + 1)
        r = r + y[i][k] * z[k][j];
    x[i][j] = r;
}
    
```

What?

How?



Focus

Potential?

Where?

Any Compiler



Sampler

Finger Print (~2MB)

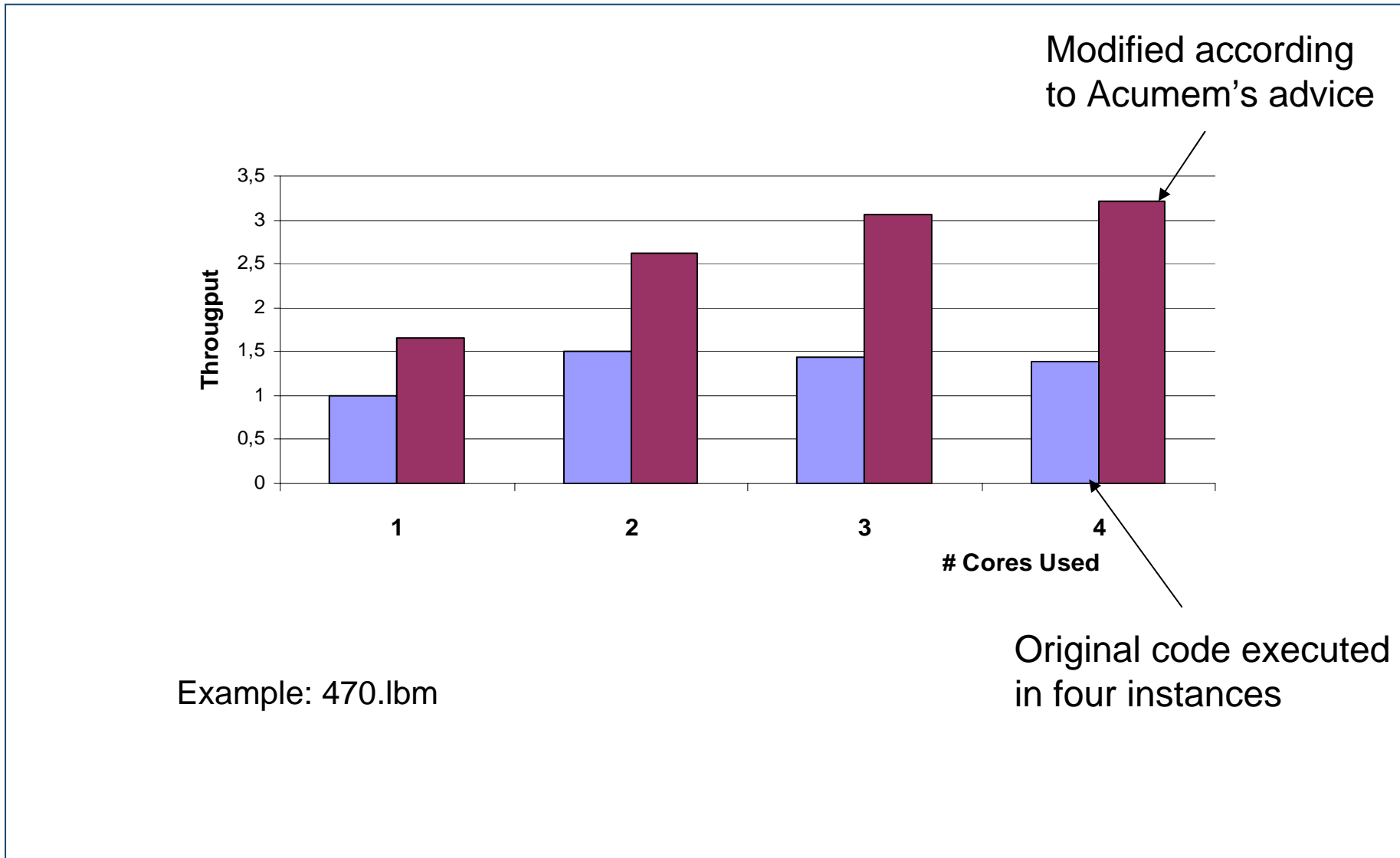
Analyzer

Target System Parameters

Host System = Linux/Solaris x86

DEMO

How Acumem can Help

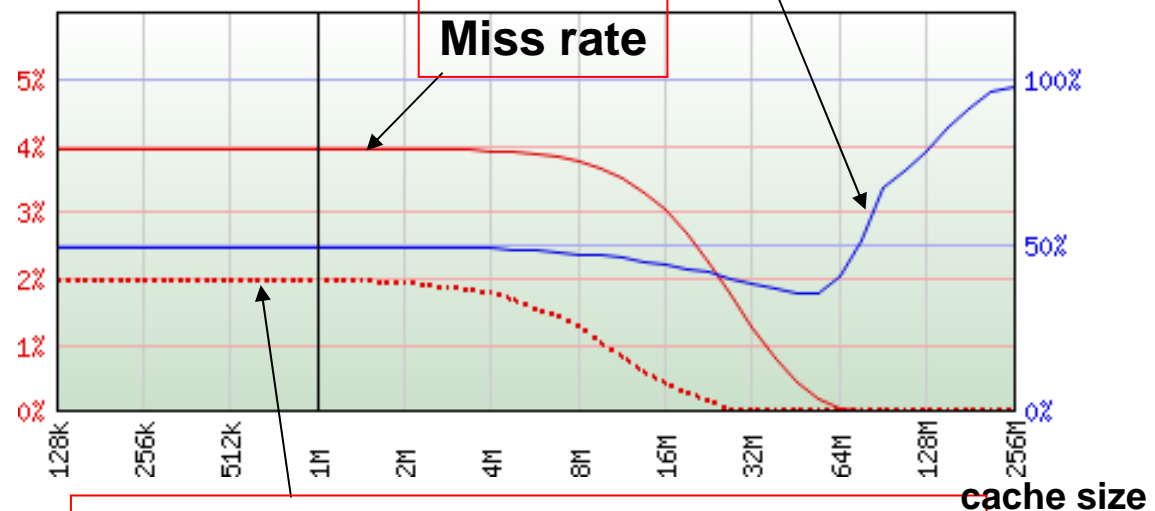


Acumem gives overall understanding (Example: 462.libquantum)



Info presented by the Acumem tool based on fingerprint data from a single run

Utilization (fraction of data in the cache actually used by the app)
Scale to the right of the graph



Predicted miss rate if utilization was fixed

Advice from Acumem VPE:

- Change a data structure from vector of *structs* to *vectors*
- Projected overall miss rate: 4.0% → 2.0% @1MB

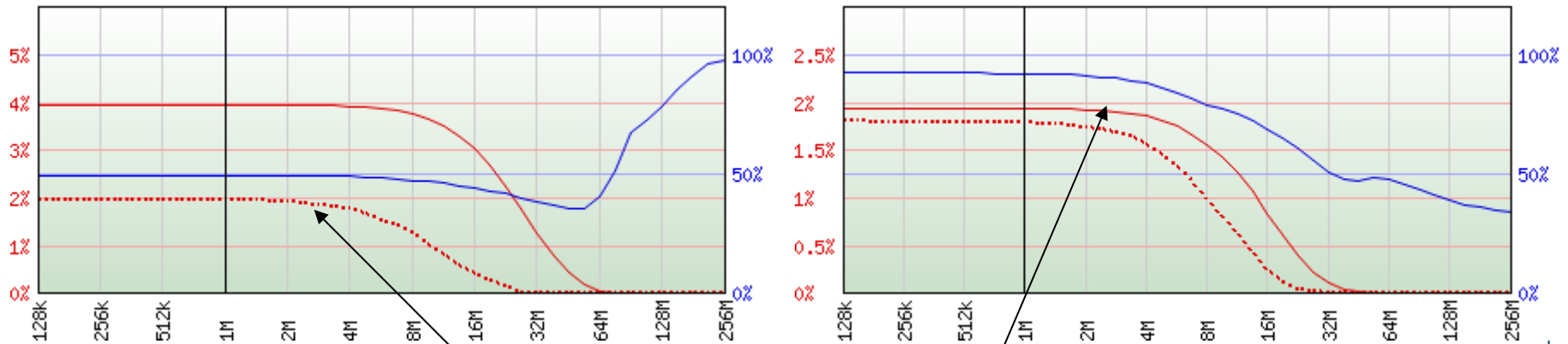
Performance curves for the modified 462.libquantum



SPEC CPU2006-462.libquantum

Before

After



Predicted miss rate close to the achieved miss rate

- Cache utilization improved from 50% to 92% at 1MB
- Miss rate cut in half (as predicted)
(And memory bandwidth cut in half !!)

Virtual Performance Expert (VPE)



Acumem VPE: ./art (64k/64) - Mozilla Firefox

Choose focus: Bandwidth, Latency or Loop

List important speed crimes

Loop / Issue	Summary	% of fetches	Utilization	HW-Prefetch	Randomness
1 / 3	Poor utilization	29.4%	12.4%	100.0%	Low
1 / 4	Loop fusion	29.4%	12.4%	97.6%	Low
1 / 1	Inefficient loop nesting	29.2%	12.6%	0.0%	Low
3 / 2	Loop fusion	4.4%	11.8%	97.3%	Low
3 / 100	Poor utilization	4.4%	23.7%	100.0%	Low

Plenty of stats gives an overview

Click for help

Issue #8: Cache line utilization

This instruction group also show symptoms of: Hot-Explanation

Statistics for instructions of this issue

% of misses	0.0%
% of fetches	4.4%
Fetch rate	51.7%
Cache line utilization	23.7%
HW prefetch probability	100.0%
Access randomness	Low
Worst instruction	match() , scanner.c:557

Guilty source line

Drill-down capabilities

Identify guilty data structure

```

543     for (j=0;j<9 && !fires ;j++)
544
545
546
547     tnorm = 0;
548     for (ti=0;ti<numfls;ti++)
549     {
550         .I[op] + a*(f1_layer[ti].U);
551
552         tnorm += f1_layer[ti].W * f1_layer[ti].W;
553     }
554     tnorm = sqrt((double)tnorm);
555     /* Compute f1 layer - X values */
556     for (tj=0;tj<numfls;tj++)
557         f1_layer[tj].X = f1_layer[tj].W/tnorm;
558
559
560     /* Co
561
562     tnorm
563     for (ti=0;ti<numfls;ti++)
564     {
565         if (f1_layer[ti].X < theta)
566
567
568         .I[op] + a*(f1_layer[ti].U);
569
570
571
572
573     tnorm += f1_layer[ti].V * f1_layer[ti].V;
    
```

Improving Throughput (libquantum)

